(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－181967

(43)公開日　平成７年(1995)７月21日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ１０Ｈ	1/00	１０１　Ｃ Ｚ				
	7/02					
			8938－5Ｈ	Ｇ１０Ｈ	7/00　５２１　Ｆ	

審査請求　未請求　請求項の数２　ＯＬ　（全 16 頁）

(21)出願番号　特願平5－328841

(22)出願日　平成５年(1993)12月24日

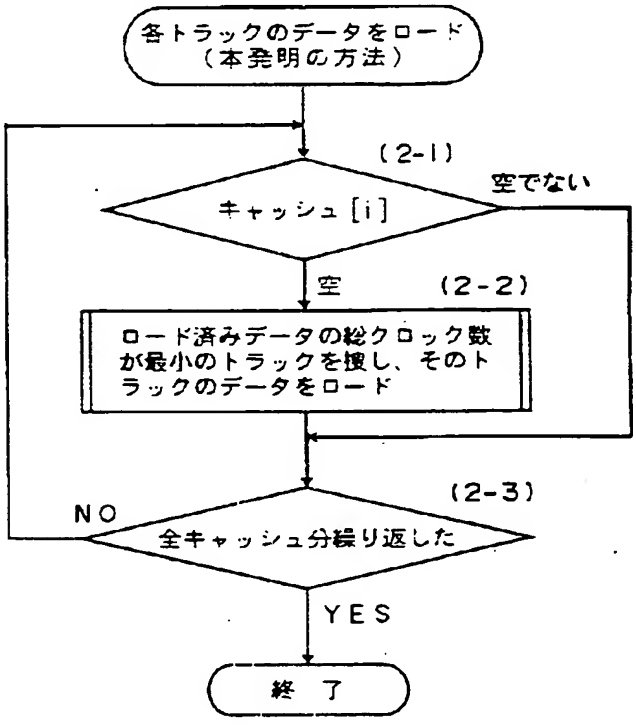
(71)出願人　000116068
ローランド株式会社
大阪府大阪市北区堂島浜１丁目４番16号

(72)発明者　佐藤　健二
大阪市北区堂島浜１丁目４番16号　ローランド株式会社内

(74)代理人　弁理士　山田　正紀　（外２名）

(54)【発明の名称】　演奏データ読込装置

(57)【要約】
【目的】本発明は、例えばフロッピィディスク等の外部メモリに記憶された演奏データを読み込む演奏データ読込装置に関し、内部メモリの使用効率を向上させる。
【構成】空いている（または空いた）キャッシュにどこのトラックの演奏データをロードするかを固定的には定めず、ロードされている演奏データの総クロック（演奏時間）の最小のトラックの演奏データをロードする。



【特許請求の範囲】

【請求項１】　装填された外部メモリから、１つの曲を構成する複数のトラックの演奏データを読み込む演奏データ読込装置において、

前記外部メモリから読み込まれた演奏データを、該外部メモリをアクセスする単位で格納する複数の内部メモリと、

前記複数の内部メモリに格納された演奏データを、各トラックそれぞれについて順次取り出すデータ取出し手段と、

前記複数の内部メモリ中に空の内部メモリが発生した際、前記複数の内部メモリに格納されている各トラック毎の演奏データのうち最短時間で演奏が終了するトラックの演奏データに続く、該トラックの次の演奏データを前記外部メモリから読み込んで空の内部メモリに格納する読込制御手段とを備えたことを特徴とする演奏データ読込装置。

【請求項２】　ディスク装置から、１つの曲を構成する複数のトラックの演奏データを読み込む演奏データ読込装置において、

前記ディスク装置から読み込まれた演奏データを、該ディスク装置をアクセスする単位で格納する複数の内部メモリと、

前記複数の内部メモリに格納された演奏データを、各トラックそれぞれについて順次取り出すデータ取出し手段と、

前記複数の内部メモリ中に空の内部メモリが発生した際、前記複数の内部メモリに格納されている演奏データの各トラック毎の演奏終了時刻と、前記ディスク装置の現在のヘッド位置とに基づいて、前記ディスク装置の、次にアクセスされるメモリ領域を定め、該メモリ領域に記憶された演奏データを前記ディスク装置から読み込んで空の内部メモリに格納する読込制御手段とを備えたことを特徴とする演奏データ読込装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、例えばフロッピィディスク等の外部メモリに記憶された演奏データを読み込む演奏データ読込装置に関する。

【０００２】

【従来の技術】近年、自動演奏装置が広く使用されている。自動演奏装置は、通常、外部メモリ、例えばフロッピィディスク（以下フロッピィディスクで代表させる）が装填可能に構成されており、好みの楽曲を表わす演奏データが記憶されたフロッピィディスクを装填して演奏を楽しむことができるように構成されている。

【０００３】このように構成された自動演奏装置において、自動演奏を行なうには、従来は装填されたフロッピィディスクに記憶された演奏データを一旦ＲＡＭ等の内部メモリに一曲分全部を転送し、その転送の後、その内部メモリから演奏データを順次読み出して演奏を行なっていた。ところがこの方法では少なくとも一曲分の演奏データを記憶できる程度の大容量の内部メモリが必要であり、近年一曲分の演奏データがますます増加する傾向にあるため、ますます大容量の内部メモリを備える必要があり、コスト上好ましくない傾向となっている。

【０００４】これを解決するため、以下に説明する手法が提案されている。図２７は、従来の手法における演奏データのロード方法を示すフローチャート、図２８はその従来の手法の説明図である。ここでは、１つの曲を構成するのに必要な演奏の各パート毎の演奏データ、例えばリズムを表わす演奏データ、メロディを表わす演奏データ、ベースを表わす演奏データ等のそれぞれを「トラック」と称する。またここでは、フロッピィディスクのアクセスの単位を「セクタ」と称する。

【０００５】図２８に示すように、この自動演奏装置では最大５トラックの演奏を行なうことができるものとし、各トラック毎に、セクタ単位のデータが格納される一時メモリであるキャッシュを、キャッシュＡとキャッシュＢの２つ用意する。そして、図２７のステップ２７＿１に示すように、各トラック毎のキャッシュＡ，Ｂが空いているかどうか判断し、空いているキャッシュがあればそのキャッシュに演奏データをロードし（ステップ２７＿２）、それを全トラックについて順次繰り返していた（ステップ２７＿３）。

【０００６】例えばトラック０～４について、各トラック専用の各２つのキャッシュＡ，Ｂを用意し、それらのキャッシュＡ，Ｂにデータを格納し、キャッシュＡのデータを１パケット単位で読み出してバッファに一旦格納し、そのバッファからクロックに応じて１メッセージ単位のデータとしてＭＩＤＩ出力する。ここで、例えばトラック０についてキャッシュＡの演奏データの読み出しが終了すると、それに引き続いてキャッシュＢの演奏データが１パケット単位で順次バッファに転送され、その間にキャッシュＡに、キャッシュＢに格納された演奏データに引き続く演奏データがフロッピィディスクから読み出されて格納される。以後同様にして、キャッシュＡとキャッシュＢが交互に用いられる。他のトラック２～５についても同様である。各トラック毎にキャッシュを２つ用意するのは、フロッピィディスクからキャッシュに演奏データを転送するのに、１回の転送あたり例えば０．２５秒程度必要とし、これはテンポ１２０で考えると８分音符１つ分程度に相当し、１つのトラックについてキャッシュを１つしか備えないとそのキャッシュに格納された演奏データによる演奏が終了した後次の演奏データが転送されるまでの間、演奏が中断してしまうからである。

【０００７】このように構成することにより、フロッピィディスクに格納された演奏データを１曲分まとめてロードすることのできる大容量の内部メモリを備えること

なく、演奏データをフロッピィディスクから少しずつ順次ロードしながら演奏を行なうことができる。

【０００８】

【発明が解決しようとする課題】上述したように１回のデータ転送に約０．２５秒程度必要とするため、上述の、フロッピィディスクから内部のキャッシュに少しずつデータ転送を行なう手法では、この転送速度よりキャッシュからデータを送出する速度の方が速い、例えばテンポの速い演奏を行う場合、クロックに正確な安定した演奏を行なうためには、１つのトラックあたりのキャッシュの数を増やす必要がある。しかし、そのように１つのトラックあたりのキャッシュの数を増やすと、キャッシュとして必要な全メモリ容量はその装置の最大トラック数に比例して膨大なものになってしまうということになる。また曲目によっては、その装置の最大トラック数よりも少ない数のトラックしか使用しない場合もあり、そのときにはその曲の演奏の間ずっと空のままのキャッシュが生じてしまい、メモリの使用効率が悪いという問題がある。

【０００９】本発明は、上記事情に鑑み、内部メモリの使用効率を向上させた演奏データ読込装置を提供することを目的とする。

【００１０】

【課題を解決するための手段】上記目的を達成する本発明の第１の演奏データ読込装置は、装填された外部メモリから、１つの曲を構成する複数のトラックの演奏データを読み込む演奏データ読込装置において、

（１）外部メモリから読み込まれた演奏データを、外部メモリをアクセスする単位で格納する複数の内部メモリ

（２）上記複数の内部メモリに格納された演奏データを、各トラックそれぞれについて順次取り出すデータ取出し手段

（３）上記複数の内部メモリ中に空の内部メモリが発生した際、それら複数の内部メモリに格納されている各トラック毎の演奏データのうち最短時間で演奏が終了するトラックの演奏データに続く、そのトラックの次の演奏データを外部メモリから読み込んで空の内部メモリに格納する読込制御手段

を備えたことを特徴とする。

【００１１】また、上記目的を達成する本発明の第２の演奏データ読込装置は、上記第１の演奏データ読込装置にいう外部メモリとしてディスク装置を備え、さらに、上記（３）の読込制御手段に代えて、

（４）上記複数の内部メモリ中に空の内部メモリが発生した際、それら複数の内部メモリに格納されている演奏データの各トラック毎の演奏終了時刻と、ディスク装置の現在のヘッド位置とに基づいて、ディスク装置の、次にアクセスされるメモリ領域を定め、そのメモリ領域に記憶された演奏データをディスク装置から読み込んで空の内部メモリに格納する読込制御手段

を備えたことを特徴とする。

【００１２】

【作用】複数のトラックの演奏データをよく観察してみると、テンポトラックのようにほとんどデータがなかったり、リズムトラックやギターデータが記録されているトラックのように大量のデータが記録されていたりと、データ量は同一ではない。

【００１３】そこで、本発明のうち第１の自動演奏装置では、空いている（または空いた）内部メモリにどこのトラックのデータをロードするかを固定的には定めず、ロードされているデータの総クロック最小（演奏時間最短）のトラックのデータをロードするようにしたため、内部メモリの利用効率が上がり、内部メモリの数が少なめであっても、過密データの、クロックに正確な演奏が可能になる。

【００１４】また、外部メモリとして、ディスク装置を備えた場合に、そのディスク装置に記憶されたデータをロードする際に、そのディスク装置のヘッドを大きく移動させるとそれだけロードする迄に時間がかかり、したがってヘッドの動きの少ない順序でロードすることが好ましい。本発明の第２の自動演奏装置は、この観点から成されたものであり、基本的には上記第１の自動演奏装置と同様ではあるが、次にロードすべきデータを、ロードされているデータの総クロックの最小トラックのデータに固定するものではなく、総クロックの小さいいくつかのデータの中からヘッドの動きが少なくて済むデータを先にロードすることにより、全体としてより迅速なロードが行なわれる。

【００１５】

【実施例】以下、本発明の実施例について説明する。図１は、本発明の演奏データ読込装置の一実施例の構成を表わすブロック図である。この自動演奏装置１０には、プログラムを実行するＣＰＵ１１、ＣＰＵ１１で実行されるプログラムを格納するＲＯＭ１２、プログラムの作業領域として使用されるＲＡＭ１３が備えられている。本実施例では、このＲＡＭ１３内部に、本発明にいう複数のキャッシュが確保される。

【００１６】またこの自動演奏装置１０には、演奏の開始（ＰＬＡＹ）、停止（ＳＴＯＰ）等を指示するスイッチ１４、この自動演奏装置１０の状態、例えば演奏中の曲の曲番号や小節番号等を表示する表示器１５、フロッピィディスク装置１６を制御してフロッピィディスク装置１６に装填されたフロッピィディスクをアクセスするフロッピィディスク制御装置１７、演奏データをＭＩＤＩ出力端子から出力するＭＩＤＩ通信装置１８が接続されている。これらの各構成要素１１～１５，１７，１８はバス１９により互いに接続されている。ＭＩＤＩ通信装置１８から出力された演奏データは、外部に接続された。例えば自動演奏装置に入力され、この自動演奏装置により、入力された演奏データに基づく自動演奏が行な

われる。尚、ここでは、演奏データ読込装置に自動演奏装置が接続されるものとして説明するが、本実施例の演奏データ読込装置を内包した自動演奏装置を構成してもよいことはもちろんである。

【００１７】図２、図３は、本実施例におけるデータ転送処理の基本概念を示した、それぞれフローチャート、データの流れを示す説明図である。図３に示すように、この実施例では５つのキャッシュが用意されており、図２のルーチンのステップ２＿１において番号ｉ（ｉ＝０～４）のキャッシュが空か否かを判断し、空のキャッシュがあった場合、ステップ２＿２において、ロード済データの総クロック数が最小のトラックを捜し、そのトラックのデータをロードする。これを全キャッシュについて繰り返す（ステップ２＿３）。すなわち、ここには、キャッシュとトラックとの固定的な対応づけはなく、空きのキャッシュが発生する毎に、総クロック数が最小、即ちロードした演奏データが最も早くなくなってしまうトラックの演奏データを、その空のキャッシュに転送するようにしたものである。

【００１８】以下、このデータ転送について詳細に説明する。図４は、本実施例における、フロッピィディスク装置１６（図１参照）に装填されたフロッピィディスク中の、今から演奏しようとするある曲の演奏データのデータ構造を示した図である。この曲の演奏には３つのトラックが使用され、トラック１の演奏データは、そのフロッピィディスクのセクタ０と、セクタ１の一部とに記憶されており、その各セクタ内のトラック０の演奏データのクロック数（演奏時間）はそれぞれ１５０，１０である。またトラック１の演奏データは、セクタ１の一部と、セクタ２，セクタ３と、セクタ４の一部とに記憶されており、その各セクタ内のトラック１のクロック数は、それぞれ、１０，５０，５０，５０である。同様に、トラック２の演奏データは、セクタ４の一部と、セクタ５，６に記憶されており、各セクタ内のトラック２のクロック数は、それぞれ２０，１００，４０である。

【００１９】図５～図１４は、フロッピィディスクに記憶された図４に示すデータ構造の演奏データが、図３に示す５つのキャッシュに転送（ロード）される際の転送手順と、そのデータ転送に伴う図１に示すＣＰＵ１１で実行されるデータ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【００２０】これらの図５～図１４に示す各種フラグ、レジスタ等の役割りは以下のとおりである。

・ＬｏａｄＰｔｒ［ｉ］：番号ｉ（ｉ＝０～２）のトラックの演奏データを最後にロードしたキャッシュの番号（０～４）を格納するレジスタ

・ＬｏａｄＴｉｍｅ［ｉ］：番号ｉのトラックの、それまでキャッシュにロードされた演奏データの総クロック数を格納するレジスタ

・ＳｅｃｔＣｎｔ［ｉ］：番号ｉのトラックの、未だキャッシュにロードされていない、残りのセクタの数を格納するレジスタ

・ＰｌａｙＮｏ［ｉ］：番号ｉのトラックの、次に出力される演奏データが格納されているキャッシュの番号を格納するレジスタ

・Ｃａｃｈｅ［ｊ］：番号ｊのキャッシュにロードされた演奏データが記憶されていた、フロッピィディスクのセクタの番号を格納するレジスタ

・Ｆｒｅｅ［ｊ］：番号ｊのキャッシュが空（‘１’）であるか否（‘０’）かを示すフラグ

・Ｎｅｘｔ［ｊ］：番号ｊのキャッシュに格納された演奏データに引き続く、その演奏データのトラックと同一のトラックの演奏データが格納されたキャッシュの番号を格納するレジスタ。自分が最後のときは‘－１’が格納される。

【００２１】以下、図５～図１４について順を追って説明する。図５、図６は、演奏開始前の演奏準備段階におけるデータ転送を示すものであり、先ず図５に示すように、最初は全てのキャッシュが空であるから、キャッシュ０，１，２に、それぞれトラック０，１，２の各先頭の演奏データが格納されたセクタ０，１，４の演奏データが順次転送される。これにより、ＬｏａｄＰｔｒ［０］，ＬｏａｄＰｔｒ［１］，ＬｏａｄＰｔｒ［２］に、それぞれ、転送先のキャッシュ番号０，１，２が格納され、ＬｏａｄＴｉｍｅ［０］，ＬｏａｄＴｉｍｅ［１］，ＬｏａｄＴｉｍｅ［２］に、それぞれ、転送されたセクタ内の各トラック０，１，２の演奏データの総クロック数‘１５０’，‘１０’，‘２０’が格納される。またＳｅｃｔＣｏｎｔ［０］，ＳｅｃｔＣｏｎｔ［１］，ＳｅｃｔＣｏｎｔ［２］には、それぞれ、トラック０，１，２の演奏データが格納された、未だキャッシュに転送されていないセクタ数‘１’，‘３’，‘２’が格納される。

【００２２】さらにＰｌａｙＮｏ［０］，ＰｌａｙＮｏ［１］，ＰｌａｙＮｏ［２］には、演奏のための各トラックの演奏データの取り出し先であるキャッシュ番号‘０’，‘１’，‘２’が格納される。またＣａｃｈｅ［０］，Ｃａｃｈｅ［１］，Ｃａｃｈｅ［２］には、そのキャッシュに転送された演奏データが記憶されていたフロッピィディスクのセクタ番号‘０’，‘１’，‘４’が格納される。キャッシュ３，４には未だ転送されていないため、Ｃａｃｈｅ［３］，Ｃａｃｈｅ［４］は空である。またキャッシュ０，１，２にはデータが転送されたため、Ｆｒｅｅ［０］～Ｆｒｅｅ［２］には、それぞれ‘０’が格納される。キャッシュ３，４は空であるためＦｒｅｅ［３］，Ｆｒｅｅ［４］には‘１’が格納されている。さらに、キャッシュ１，２，３に格納された演奏データが、現在のところ、各トラック０，１，２について転送された演奏データの最終であるた

め、Ｎｅｘｔ［0］～Ｎｅｘｔ［2］には、最終であることを示す‘－1’が格納され、また、Ｎｅｘｔ［3］，Ｎｅｘｔ［4］には初期値としての‘－1’が格納されている。

【００２３】次に、ＬｏａｄＴｉｍｅ［0］～ＬｏａｄＴｉｍｅ［2］を比較し、総クロック数の最小のトラック（ここではトラック1）について空のキャッシュ3に演奏データが転送され、空のキャッシュ4がまだ残っているため、キャッシュ3にデータが転送された段階における、ＬｏａｄＴｉｍｅ［0］～ＬｏａｄＴｉｍｅ［2］を比較し、総クロック数の最小のトラック（今度はトラック2）について空のキャッシュ4に演奏データが転送され、これにより、キャッシュ0～4の全てについて演奏データが転送される。

【００２４】図6は、そのときの状態を示した図である。ＬｏａｄＰｔｒ［1］，ＬｏａｄＰｔｒ［2］には、それぞれ、対応するトラックの、最後に転送したデータのデータ転送先であるキャッシュ番号‘3’，‘4’が格納され、ＬｏａｄＴｉｍｅ［1］，ＬｏａｄＴｉｍｅ［2］には、それぞれ、トラック1，2に関する転送された演奏データの総クロック数‘10＋50’，‘20＋12’0が格納され、ＳｅｃｔＣｏｎｔ［1］，ＳｅｃｔＣｏｎｔ［2］には、それぞれ各トラック1，2のデータ未転送のセクタ数‘2’，‘1’が格納される。

【００２５】また、Ｃａｃｈｅ［3］，Ｃａｃｈｅ［4］には、そこに格納された演奏データのセクタ番号‘2’，‘5’が格納され、Ｆｒｅｅ［3］，Ｆｒｅｅ［4］にはデータが格納されていることを示す‘0’が格納される。さらに、Ｎｅｘｔ［1］，Ｎｅｘｔ［2］には、次の演奏データの格納先であるキャッシュ番号‘3’，‘4’が格納される。

【００２６】このようにして、全てのキャッシュ0～4に演奏データがロードされた後、演奏が開始される。図7は、演奏開始後、10クロックに相当する時間が経過した後の状態を表わすものである。10クロック経過したことにより、ＬｏａｄＴｉｍｅ［0］，ＬｏａｄＴｉｍｅ［1］，ＬｏａｄＴｉｍｅ［2］は、図6の状態のときよりもそれぞれ10クロックずつ減り、それぞれ‘140’，‘0＋50’，‘10＋100’となる。このとき、キャッシュ1に格納されていた演奏データによる演奏が終了し、Ｆｒｅｅ［1］＝1，Ｎｅｘｔ［1］＝－1に変更され、ＰｌａｙＮｏ［1］＝3に変更される。

【００２７】これによりキャッシュ1は空になったため、ＳｅｃｔＣｎｔ［0］～ＳｅｃｔＣｎｔ［2］が‘0’ではないトラック（ここではトラック0～2のいずれも‘0’ではない）に関し、ＬｏａｄＴｉｍｅ［0］～ＬｏａｄＴｉｍｅ［2］の内容が比較され、総クロック数の最小のトラック1の、次の演奏データがキャッシュ1にロードされる。

【００２８】図8は、キャッシュ1にトラック1の演奏データがロードされた後の状態を示した図である。キャッシュ1にセクタ3の演奏データが転送され、Ｃａｃｈｅ［1］＝3，Ｆｒｅｅ［1］＝0に変更され、Ｎｅｘｔ［3］＝1に変更され、さらにＬｏａｄＰｔｒ［1］＝1，ＬｏａｄＴｉｍｅ［1］＝50－50，ＳｅｃｔＣｏｎｔ［1］＝1に変更される。

【００２９】図9は、図8の状態から10クロック経過後の状態を示す図である。10クロック経過したことにより、ＬｏａｄＴｉｍｅ［0］，ＬｏａｄＴｉｍｅ［1］，ＬｏａｄＴｉｍｅ［2］は、図8の状態のときよりもそれぞれ10クロックずつ減り、‘130’，‘40＋50’，‘0＋100’となる。このときキャッシュ2に格納された演奏データによるトラック2の演奏が終了し、Ｆｒｅｅ［2］＝1，Ｎｅｘｔ［2］＝－1に変更される。またＰｌａｙＮｏ［2］が4に変更され、キャッシュ4に格納されたトラック2の演奏データによる演奏が開始される。

【００３０】また、キャッシュ2は空になったため、ＳｅｃｔＣｎｔ［0］～ＳｅｃｔＣｎｔ［2］が‘0’ではないトラックに関しＬｏａｄＴｉｍｅ［0］～ＬｏａｄＴｉｍｅ［2］の内容が比較され、総クロック数の最小のトラック1の、次の演奏データが、図10に示すように、キャッシュ2にロードされる。キャッシュ2に、トラック1の次の演奏データ、即ちセクタ4の演奏データが転送され、Ｆｒｅｅ［2］＝0に変更され、Ｎｅｘｔ［1］＝2に変更され、さらにＬｏａｄＰｔｒ［1］＝2，ＬｏａｄＴｉｍｅ［1］＝40＋50＋50，ＳｅｃｔＣｎｔ［1］＝0に変更される。

【００３１】ＳｅｃｔＣｎｔ［1］＝0となったことから、トラック1の演奏データは、全てキャッシュにロードされたことになる。図11は、図10に示す状態から、さらに40クロック経過後の状態を示す図である。40クロック経過したことにより、ＬｏａｄＴｉｍｅ［0］，ＬｏａｄＴｉｍｅ［1］，ＬｏａｄＴｉｍｅ［2］は、図10の状態よりもそれぞれ40クロックずつ減り、‘90’，‘0＋50＋50’，‘60’となる。

【００３２】このときキャッシュ3に格納されたトラック1の演奏データは空になり、Ｆｒｅｅ［3］＝1，Ｎｅｘｔ［3］＝－1に変更され、ＰｌａｙＮｏ［1］＝1に変更され、今度はキャッシュ1に格納されたトラック1の演奏データによる演奏が行なわれる。またキャッシュ3は空になったため、ＳｅｃｔＣｎｔ［0］～ＳｅｃｔＣｎｔ［2］が‘0’ではないトラックに関し、ＬｏａｄＴｉｍｅ［0］～ＬｏａｄＴｉｍｅ［2］の内容が比較され、総クロック数の最小のトラック2の、次の演奏データがキャッシュ3にロードされる。

【００３３】図12は、キャッシュ3にトラック2の、

次の演奏データ、即ちセクタ６の演奏データがロードされた状態を示した図である。Ｃａｃｈｅ［3］＝6，Ｆｒｅｅ［3］＝0に変更され、また、Ｎｅｘｔ［4］＝3に変更される。さらに、ＬｏａｄＰｔｒ［2］＝3，ＬｏａｄＴｉｍｅ［2］＝６０＋４０，ＳｅｃｔＣｎｔ［2］＝0に変更される。

【００３４】図１３は、図１２の状態から更に５０クロック経過した後の状態を示す図である。５０クロック経過したことにより、ＬｏａｄＴｉｍｅ［0］，ＬｏａｄＴｉｍｅ［1］，ＬｏａｄＴｉｍｅ［2］は、図１２に示す状態のときよりもそれぞれ５０クロックずつ減少し、それぞれ、４０，０＋１５０，１０＋４０となる。

【００３５】このとき、キャッシュ１に格納されたトラック１の演奏データは空になり、Ｆｒｅｅ［1］＝1，Ｎｅｘｔ［1］＝－1に変更され、ＰｌａｙＮｏ［1］＝2に変更される。また、キャッシュ１が空になったことから、ＳｅｃｔＣｎｔ［0］～ＳｅｃｔＣｎｔ［2］が‘0’ではないトラックに関し、ＬｏａｄＴｉｍｅ［0］～ＬｏａｄＴｉｍｅ［2］が比較されるが、ここではＳｅｃｔＣｎｔ［1］＝0，ＳｅｃｔＣｎｔ［2］＝0、すなわち、トラック１，２の演奏データのキャッシュへのロードは既に終了しているため、残りのトラック０の演奏データがキャッシュ１にロードされる。

【００３６】図１４は、キャッシュ１にトラック０の次の演奏データ、即ちセクタ１の演奏データが転送された状態を示した図である。Ｃａｃｈｅ［1］＝1，Ｆｒｅｅ［1］＝0に変更され、またＮｅｘｔ［0］＝1に変更される。さらにＬｏａｄＰｔｒ［0］＝1，ＬｏａｄＴｉｍｅ［0］＝４０＋１０，ＳｅｃｔＣｎｔ［0］＝0に変更される。

【００３７】その後１０クロック経過すると、ＬｏａｄＴｉｍｅ［0］＝３０＋１０，ＬｏａｄＴｉｍｅ［1］＝４０，ＬｏａｄＴｉｍｅ［2］＝0＋４０となって、トラック２の、キャッシュ４に格納された演奏データによる演奏が終了し、Ｆｒｅｅ［4］＝1，Ｎｅｘｔ［4］＝－1，ＰｌａｙＮｏ［2］＝3に変更される。さらにその後３０クロック経過すると、ＬｏａｄＴｉｍｅ［0］＝0＋１０，ＬｏａｄＴｉｍｅ［1］＝１０，ＬｏａｄＴｉｍｅ［2］＝１０となってキャッシュ０に格納されたトラック０の演奏データが空になって、Ｆｒｅｅ［0］＝1，Ｎｅｘｔ［0］＝－1，ＰｌａｙＮｏ＝1に変更される。

【００３８】さらにその後ろ１０クロック経過すると、この曲一曲分の演奏データによる演奏が全て終了する。以下、フロッピィディスクに記憶された演奏データのデータフォーマットについて説明し、次いで、図５～図１４を参照して説明した演奏データの転送に関連した各種ルーチンについて説明する。

【００３９】図１５は、フロッピィディスクに記憶された演奏データのフォーマットを示した図である。先頭に１バイトのヘッダがありそこに総トラック数が記述されている。その後ろは、トラック毎のデータとなっており、トラック毎のデータの先頭の４バイトはそのトラックの長さ（バイト数）が記述され、その後ろに実際の演奏データが記述されている。

【００４０】図１６は演奏データのフォーマットを示した図である。各演奏データは２バイトの時間情報部分と３バイトのデータ情報部分で構成されている。図１７は演奏データ中のデータ情報部分の詳細を示した図である。３バイトのＭＩＤＩ情報はそのまま記述され、ＭＩＤＩ情報が２バイト、１バイトの場合は余分なところをＦＦｈで埋めている。さらに、時間情報のみ、テンポ、拍子、終了情報なども記述することができる。

【００４１】次に、図１８～図２６のルーチンについて説明するが、既に図５～図１４を参照してデータ転送の具体例について詳細に説明したため、以下では、各ルーチンについて簡単に説明するにとどめる。以下に、既出のものも含め、図１８～図２６のルーチンで使用される主なレジスタ、フラグの役割りについてまとめて説明する。

・Ｃａｃｈｅ［MAX＿CACHE］［５１２］：ディスクから演奏データがロードされるキャッシの番号が格納されるレジスタ

・ＰｌａｙＮｏ［MAX＿ＴＲＡＣＫ］：次に出力される演奏データが格納されたキャッシュの番号が格納されるレジスタ

・ＰｌａｙＡｄｄｒ［MAX＿ＴＲＡＣＫ］：次に出力される演奏データのキャッシュ内のアドレスが格納されるレジスタ

・ＴｉｍｅＣｏｕｎｔ［MAX＿ＴＲＡＣＫ］：待時間カウンタ

・ＬｏａｄＰｔｒ［MAX＿ＴＲＡＣＫ］：最後にロードされたキャッシュの番号が格納されるレジスタ

・ＬｏａｄＴｉｍｅ［MAX＿ＴＲＡＣＫ］：既にキャッシュにロードされた演奏データの、トラック毎のクロック数を記憶するレジスタ

・ＳｅｃｔＣｎｔ［MAX＿ＴＲＡＣＫ］：各トラックのロードが終了したか調べるために演奏データが記録されている残りのセクタ数が格納されるレジスタ

・ＥｎｄＦｌａｇ［MAX＿ＴＲＡＣＫ］：そのトラックの演奏が既に終了したか否かを示すフラグ

・Ｎｅｘｔ［MAX＿CACHE］：次のキャッシュ番号が格納されるレジスタ

・Ｆｒｅｅ［MAX＿CACHE］：そのキャッシュが空か否かを示すレジスタ（‘1’が空を表わす）

・ＳｅｔＤａｔａ［ｔｒｋ］［3］：出力される演奏データをトラック毎に記憶するレジスタ

・ｗｏｒｋ［5］：５バイトの演奏データを時間情報部分とデータ情報部分とに分離するために使用される作業用レジスタ

・ｗｏｒｋ０：ロード済み演奏データの総クロック数が最小のトラックを捜すために総クロック数を一時的に格納しておく作業用レジスタ
・Ｓｏｎｇ：曲番号が格納されるレジスタ
・ＥｎｄＣｏｕｎｔ：曲の終了を調べるためのレジスタ
・Ｂｕｆｆ：総トラック数を調べるための作業用レジスタ
・ＴｒｋＮｏ：その曲のトラック数が格納されるレジスタ
・ｔｒｋ：トラックループカウンタ１
・ｔｒｋ０：トラックループカウンタ２
・ｔｒｋ１：ロード済クロック（ＬｏａｄＴｉｍｅ）が最小のトラックのトラック番号が格納されるレジスタ
・ｉ：ループカウンタ
・ＭｅａｓＣｏｕｎｔ：小節番号計測用レジスタ
・Ｍｅａｓｕｒｅ：小節番号が格納されるレジスタ
図１８は電源オンで実行が開始される、メインルーチンのフローチャートである。

【００４２】ステップ１８＿１では、フロッピィディスク装置にフロッピィディスクが入っているか否かが調べられる。尚、図中には示されていないが、フロッピィディスクが抜かれると、クロック停止と全音楽消去の処理が行なわれた後、ステップ１８＿１に飛ぶ。ステップ１８＿２では、曲番号を記憶しておくレジスタＳｏｎｇに‘０’がセットされる。本実施例では、簡単のため、曲の演奏は必ず０番の曲から開始されるものとしてステップ１８＿３では、初期化が行なわれる。すなわち、全トラック分のＰｌａｙＮｏ，ＰｌａｙＡｄｄｒ，ＴｉｍｅＣｏｕｎｔ，ＬｏａｄＰｔｒ，ＳｅｃｔＣｎｔ，ＬｏａｄＴｉｍｅに‘０’，ＥｎｄＦｌａｇに‘１’、全キャッシュのＮｅｘｔに‘０’、Ｆｒｅｅに‘１’、ＭｅａｓＣｏｕｎｔ，Ｍｅａｓｕｒｅに‘０’が格納される。

【００４３】ステップ１８＿４では、曲番号記憶レジスタＳｏｎｇで示される曲の演奏準備が行なわれる。詳細については後述する。ステップ１８＿５では、小節カウンタの準備が行なわれる。詳細は後述する。以上で演奏の準備が終了し、ステップ１８＿６で、プレイ（ＰＬＡＹ）スイッチが押されるのを待つ。

【００４４】ＰＬＡＹスイッチが押されるとステップ１８＿７に進み、クロック割り込みの発生が指示される。ステップ１８＿８ではストップ（ＳＴＯＰ）スイッチが押されたか否かが調べられ、押されていないときはステップ１８＿９に進んで曲の終了を表わすレジスタＥｎｄＣｏｕｎｔが調べられ、曲の途中（ＥｎｄＣｏｕｎｔ＞０）のときはステップ１８＿１０に進む。

【００４５】ステップ１８＿１０では、ディスクからキャッシュ（Ｃａｃｈｅ）へのデータロードが指示され、ステップ１８＿１１では小節番号表示が指示される。ステップ１８＿１０，１８＿１１の詳細については後述する。一方、ＳＴＯＰスイッチが押された場合、もしくは曲の終了の場合、ステップ１８＿１２に進み、ＳＴＯＰスイッチが押されたことに伴う処理が指示される。詳細については後述する。

【００４６】ステップ１８＿１３ではＳｏｎｇに１が加えられ、ステップ１８＿４ではＳｏｎｇがフロッピィディスクに記録されている曲数以下か否か調べられ、フロッピィディスク中にまだ演奏されていない曲が残っているときはそのままステップ１８＿３に戻り、最終曲まで演奏されたときはＳｏｎｇに‘０’がセットされて先頭の曲に戻った上でステップ１８＿３に戻る。

【００４７】以上のようにして、フロッピィディスクに記憶された演奏データに基づいて、循環的に演奏が繰り返される。図１９は、図１８のステップ１８＿４で実行される、Ｓｏｎｇで示される曲の演奏準備を行なうサブルーチンのフローチャートである（図５参照）。先ずステップ１９＿１では、その曲のトラック数を調べるために、その曲ファイル（図１５参照）の先頭部分の１セクタを、作業用レジスタＢｕｆｆにロードする。

【００４８】ステップ１９＿２では、その曲のトラック数を記憶するレジスタＴｒｋＮｏ、曲の終了を調べるためのレジスタＥｎｄＣｏｕｎｔに、ヘッダに書かれているトラック数をセットする。ステップ１９＿３では、トラックループカウンタｔｒｋに‘０’をセットする。

【００４９】ステップ１９＿４では、ロードの終了を検出するために、ｔｒｋトラックで示されるのデータが何セクタに渡って記録されているか計算し、その値をＳｅｃｔＣｎｔ［ｔｒｋ］にセットする。ステップ１９＿５では、ｔｒｋで示されるトラックの先頭の演奏データが格納されているセクタの演奏データをＣａｃｈｅ［ｔｒｋ］にロードする。

【００５０】ステップ１９＿６では、ステップ１９＿５でロードが発生したので、ＳｅｃｔＣｎｔ［ｔｒｋ］から１が引き算される。ステップ１９＿７では、ステップ１９＿５で何クロック分の演奏データがロードされたかを調べ、そのクロック数を、キャッシュにこれまでにロードされていた演奏データのクロック数を記憶するレジスタＬｏａｄＴｉｍｅ［ｔｒｋ］にセットする。

【００５１】ステップ１９＿８では、最後にロードされたキャッシュの番号を記憶するレジスタＬｏａｄＰｒｔ［ｔｒｋ］に、ｔｒｋをセットする。ステップ１９＿９では、次に出力される演奏データが格納されたＣａｃｈｅの番号を記憶するレジスタＰｌａｙＮｏ［ｔｒｋ］にｔｒｋをセットし、さらに、次に出力される演奏データのキャッシュ内のアドレスを記憶するレジスタＰｌａｙＡｄｄｒ［ｔｒｋ］に演奏データの先頭アドレスをセットする。

【００５２】ステップ１９＿１０では、次の演奏データのセットが指示される。詳細については後述する。ステップ１９＿１１では、ｔｒｋに‘１’が加えられ、ステップ１９＿１２ではｔｒｋがＴｒｋＮｏ以内か否か調べ

られ、ｔｒｋがＴｒｋＮｏ以内であればステップ１９＿４に戻り、ｔｒｋがＴｒｋＮｏを越えるとステップ１９＿１３へ進む。ステップ１９＿１３では、図１８に示すメインルーチンにおけるステップ１８＿１０と同様に、演奏データのロードが指示される。ステップ１９＿１３のルーチンの詳細については後述する。

【００５３】図２０は、図１９のルーチンのステップ１９＿１０で実行される、次の演奏データをセットする処理を行なうサブルーチンのフローチャートである（図６参照）。ステップ２０＿１では、ＰｌａｙＮｏ［ｔｒｋ］が作業レジスタｎｏにセットされ、また、ＰｌａｙＡｄｄｒ［ｔｒｋ］が作業レジスタａｄｄｒにセットされる。

【００５４】ステップ２０＿２では、作業用ループカウンタｉに‘０’がセットされる。ステップ２０＿３では、ｉが５以内か否か調べられる。これは、演奏データは、図１６にあるように５バイトで構成されており、１バイトずつ読み込むため５回読み込む必要があるからである。ｉ＜５の場合ステップ２０＿４に進み、ａｄｄｒが５１２（１セクタのバイト数）以内か否か調べられる。ａｄｄｒ≧５１２の場合、ステップ２０＿５に進み、ＰｌａｙＮｏ［ｔｒｋ］に、次に演奏される演奏データがロードされたキャッシュ番号が記憶されたレジスタＮｅｘｔ［ｎｏ］がセットされ、ＰｌａｙＡｄｄｒ［ｔｒｋ］に‘０’がセットされ、Ｎｅｘｔ［ｎｏ］に、次が存在しないことを示す値‘－１’がセットされ、さらにＦｒｅｅ［ｎｏ］にそのキャッシュが空であることを示す値‘１’がセットされる。

【００５５】ステップ２０＿６では、作業用レジスタｗｏｒｋ［ｉ］にキャッシュＣａｃｈｅ［ｎｏ］［ａｄｄｒ］がセットされ、ステップ２０＿７でｉに‘１’が加えられてステップ２０＿３に戻る。ステップ２０＿３でｉ≧５であると判定されるとステップ２０＿８に進み、待時間カウンタＴｉｍｅＣｏｕｎｔ［ｔｒｋ］に新しい時間が加えられる。

【００５６】ステップ２０＿９では、出力される演奏データを記憶するレジスタＳｅｔＤａｔａに新しいデータがセットされる。図２１は、図１８に示すメインルーチンのステップ１８＿１０、および図１９に示すサブルーチンのステップ１９＿１３で実行される。空のキャッシュをさがしてその空のキャッシュにロード済み時間の最も少ないトラックの演奏データをロードするロードデータサブルーチンのフローチャートである。

【００５７】先ずステップ２１＿１では、キャッシュの番号を示す作業用カウンタｉに‘０’がセットされる。ステップ２１＿２では、そのキャッシュが空かどうかを示すレジスタＦｒｅｅ［ｉ］が調べられ、空いていればステップ２１＿２に進み、空いていなければステップ２１＿１４に進む。

【００５８】以下のステップ２１＿３～ステップ２１＿８はロード済み時間の少ないトラックを捜す処理である。ステップ２１＿３では、トラック番号を示す作業用レジスタｔｒｋ０に‘０’がセットされ、ロード済クロックが最小のトラックのトラック番号を示す作業用レジスタｔｒｋ１に０がセットされ、さらに最小クロック数を捜すための作業用レジスタｗｏｒｋ０に、本装置で扱える最大時間ＭＡＸ＿ＴＩＭＥがセットされる。

【００５９】ステップ２１＿４では、ロード完了レジスタＳｅｃｔＣｎｔ［ｔｒｋ０］が正か否かが調べられ、正であればステップ２１＿５に進み、そうでなければステップ２１＿７に進む。ステップ２１＿５ではｗｏｒｋ０がＬｏａｄＴｉｍｅ［ｔｒｋ０］を越えるか否かが調べられ、ｗｏｒｋ０＞ＬｏａｄＴｉｍｅ［ｔｒｋ０］の場合ステップ２１＿６に進み、そうでないときはステップ２１＿７に進む。

【００６０】ステップ２１＿６では、作業用レジスタｗｏｒｋ０にＬｏａｄＴｉｍｅ［ｔｒｋ０］がセットされ、さらにｔｒｋ１にｔｒｋ０がセットされる。ステップ２１＿７ではｔｒｋ０に‘１’が加えられる。ステップ２１＿８では、ｔｒｋ０がＴｒｋＮｏ以下か否か調べられ、ｔｒｋ＜ＴｒｋＮｏの場合ステップ２１＿４に戻り、ｔｒｋ≧ＴｒｋＮｏの場合ステップ２１＿９に進む。

【００６１】ステップ２１＿９では、ｗｏｒｋ０がステップ２１＿３で格納されたＭＡＸ＿ＴＩＭＥから変更されたか調べられ、変更されていないときは終了し、変更されていたときはステップ２１＿１０に進む。ステップ２１＿１０以降は、データのロードに伴う処理である。ステップ２１＿１０では、ｔｒｋ１で示されるトラックの、次の演奏データがＣａｃｈｅ［ｉ］にロードされる。

【００６２】ステップ２１＿１１では、ＳｅｃｔＣｎｔ［ｔｒｋ１］から‘１’が引き算される。ステップ２１＿１２では、ステップ２１＿１０で何クロック分の演奏データがロードされたかを調べ、その値がＬｏａｄＴｉｍｅ［ｔｒｋ１］に加算される。ステップ２１＿１３では、ロードに関するリンク情報の更新のため、最後にロードされたキャッシュの番号を記憶するＬｏｎｄＰｔｒ［ｔｒｋ１］がＮｅｘｔ［ｉ］にセットされ、ＬｏｎｄＰｔｒ［ｔｒｋ１］に１がセットされ、Ｆｒｅｅ［ｉ］に‘０’がセットされる。

【００６３】ステップ２１＿１４ではｉに‘１’が加えられる。ステップ２１＿１５では、ｉがキャッシュの数ＭＡＸ＿ＣＡＣＨＥ以内か否か調べられ、未だ処理の行なわれていないキャッシュがあるときはステップ２１－２に進み、全てのキャッシュについて処理が終了したときはこのルーチンを抜ける。

【００６４】図２２は、演奏データの出力指示、次のデータのセット、ロード済みカウンタの更新、曲の終りを検出して自動的に演奏終了する指示を行なうクロック割

込みルーチンのフローチャートである。このクロック割込みルーチンは、図１８のメインルーチンのステップ１８＿７でクロック割込みの発生が指示されたことを受けて発生する割込み用クロックにより、所定時間間隔毎に起動される。

【００６５】このルーチンが起動されると、先ずステップ２２＿１において、トラックカウンタｔｒｋに‘０’がセットされる。ステップ２２＿２では、そのトラックの演奏が既に終了しているか否かを示すレジスタＥｎｄＦｌａｇ［ｔｒｋ］が調べられ、終了しているときはステップ２２＿８に進み、そうでないときはステップ２２＿３に進む。

【００６６】ステップ２２＿３では、ＴｉｍｅＣｏｕｎｔ［ｔｒｋ］から‘１’が引き算される。ステップ２２＿４では、ＴｉｍｅＣｏｕｎｔ［ｔｒｋ］が正か０以下かが調べられ、正の場合ステップ２２＿７に進み、０以下の場合ステップ２２＿５に進む。

【００６７】ステップ２２＿５ではデータの出力が指示される。このステップ２２＿５の詳細については後述する。ステップ２２＿６では、次のデータのセットが指示される。詳細については、図２０が参照され既に説明されている。

【００６８】ステップ２２＿７では、ＬｏａｄＴｉｍｅ［ｔｒｋ］から‘１’が引き算される。ステップ２２＿８では、ＥｎｄＣｏｕｎｔが正か０か調べられ、正の場合ステップ２２＿１０に進み、０の場合ステップ２２＿９に進む。ステップ２２＿９では、自動終了のために演奏停止処理が指示される。詳細については後述する。

【００６９】ステップ２２＿１０ではｔｒｋに‘１’が加算される。ステップ２２＿１１ではｔｒｋがＴｒｋＮｏ以下か否か調べられ、ｔｒｋ＜ＴｒｋＮｏのときはステップ２２＿２に戻り、ｔｒｋ≧ＴｒｋＮｏのときはステップ２２＿１２に進む。ステップ２２＿１２では、小節番号計測用レジスタＭｅａｓＣｏｕｎｔに‘１’が加算される。

【００７０】ステップ２２＿１３では、ＭｅａｓＣｏｕｎｔが１６以上か否か調べられる。ＭｅａｓＣｏｕｎｔ＞１６のときはステップ２２＿１４に進み、そうでなければそのまま終了する。ステップ２２＿１４では、ＭｅａｓＣｏｕｎｔに‘１’がセットされ、小節番号を記憶するレジスタＭｅａｓｕｒｅに‘１’が加算される。

【００７１】図２３は、図２２に示すクロック割込みルーチンのステップ２２＿５で実行される、演奏データをその種類によりＭＩＤＩ出力端子に出力したり、テンポ、拍子を更新したり、曲の終りを検出するサブルーチンのフローチャートである。ステップ２３＿１ではＤａｔａＳｅｔ［ｔｒｋ］がＭＩＤＩ情報であるか否か調べられ、ＭＩＤＩ情報であった場合はステップ２３＿２に進んで、ＤａｔａＳｅｔ［ｔｒｋ］がＭＩＤＩ出力端子から出力される。

【００７２】ステップ２３＿３では、ＤａｔａＳｅｔ［ｔｒｋ］がテンポ情報であるか否か調べられ、テンポ情報であった場合はステップ２３＿４に進みテンプが変更される。ステップ２３＿５では、ＤａｔａＳｅｔ［ｔｒｋ］が拍子情報であるか否か調べられ、拍子情報であった場合はステップ２３＿６に進み、拍子が変更される。

【００７３】ステップ２３＿７ではＤａｔａＳｅｔ［ｔｒｋ］が終了情報であるか否かが調べられ、終了情報であった場合はステップ２３＿８に進み、ＥｎｄＣｏｕｎｔから‘１’が引き算され、ＥｎｄＦｌａｇ［ｔｒｋ］に‘０’がセットされる。図２４は、図１８に示すメインルーチンのステップ１８＿１２で実行される、演奏停止処理を行なうサブルーチンのフローチャートである。

【００７４】ステップ２４＿１ではクロック割り込みが停止され、ステップ２４＿２では、発音中の全楽音の消音処理が実行される。図２５は、図１８に示すメインルーチンのステップ１８＿５で実行される、小節カウンタの曲頭の休符区間を検出し、楽譜通りの小節番号にするための処理を行なうサブルーチンのフローチャートである。

【００７５】ステップ２５＿１では、曲の先頭から最初の発音までの時間（休符時間）が検出され、その値が作業用レジスタＲｅｓｔＴｉｍｅにセットされる。ステップ２５＿２では、ＲｅｓｔＴｉｍｅが３未満か否か調べられ、３未満であればステップ２５＿３に進み、そうでなければステップ２５＿５に進む。ステップ２５＿３では、Ｍｅａｓｕｒｅに‘０’がセットされてこのルーチンを抜ける。

【００７６】ステップ２５＿４では、ＲｅｓｔＴｉｍｅが５未満か否か調べられ、５未満であればステップ２５＿５に進み、そうでなければステップ２５＿６に進む。ステップ２５＿５では、アウフタクトの小節番号として、Ｍｅａｓｕｒｅに‘－１’がセットされ、このルーチンを抜ける。ステップ２５＿６では、Ｍｅａｓｕｒｅに‘０’がセットされ、このルーチンを抜ける。

【００７７】図２６は、図１８のメインルーチンのステップ１８＿１１で実行される、小節番号を表示する処理を行なうサブルーチンのフローチャートである。ステップ２６＿１では、Ｍｅａｓｕｒｅが０以上か否か調べられ、０以上の場合ステップ２６＿２に進み、そうでなければ２６＿３に進む。ステップ２６＿２では、表示器にＭｅａｓｕｒｅ＋１の値が表示される。

【００７８】ステップ２６＿３では、表示器にＭｅａｓｕｒｅの値が表示される。尚、上記実施例では、２つのトラックに跨る演奏データが記憶されたセクタは、各トラック毎に２回ロードされているが、別のトラックで既にロードされている演奏データがどこかのキャッシュにあったら、その演奏データの再度のロードは行わず、あたかもそのキャッシュにロードしたかのようにしてロー

ドの時間を少なくするようにしてもよい。また上記実施例では、単純に、クロック数の少ないトラックの演奏データをロードするように構成されているが、空いているキャッシュにどこのトラックの演奏データをロードするかを決定する処理（図１８のステップ１８＿３～ステップ１８＿８の処理）を変更して、ロード済み総クロック数だけではなく、ディスクのヘッド移動時間等も加味した評価により、どのトラックの演奏データをロードするかを決定してもよい。

【００７９】

【発明の効果】以上説明したように、本発明の演奏データ読込装置は、空いている（または空いた）内部メモリにどこのトラックの演奏データをロードするかを固定的には定めず、ロードされている演奏データの総クロックの最小のトラックの演奏データもしくはこれにディスク装置のアクセスのためのヘッド位置を加味したトラックの演奏データをロードするものであるため、内部メモリの利用効率が上がり、少ない数の内部メモリでクロックに正確な演奏が可能となる。

【図面の簡単な説明】

【図１】本発明の演奏データ読込装置の一実施例の構成を表わすブロック図である。

【図２】本実施例におけるデータ転送処理の基本概念を示したフローチャートである。

【図３】本実施例におけるデータ転送処理の基本概念を示したデータの流れを示す説明図である。

【図４】フロッピィディスク中の、今から演奏しようとするある曲の演奏データのデータ構造を示した図である。

【図５】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図６】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図７】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図８】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図９】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図１０】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図１１】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図１２】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図１３】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図１４】データ転送のためのルーチンで使用される各種フラグ、レジスタ等の値の変化を示した模式図である。

【図１５】フロッピィディスクに記憶された演奏データのフォーマットを示した図である。

【図１６】演奏データのフォーマットを示した図である。

【図１７】演奏データ中のデータ情報部分の詳細を示した図である。

【図１８】メインルーチンのフローチャートである。

【図１９】Ｓｏｎｇで示される曲の演奏準備を行なうサブルーチンのフローチャートである。

【図２０】次の演奏データをセットする処理を行なうサブルーチンのフローチャートである。

【図２１】トラックの演奏データをロードするロードデータサブルーチンのフローチャートである。

【図２２】クロック割込みルーチンのフローチャートである。

【図２３】演奏データをその種類によりＭＩＤＩ出力端子に出力したり、テンポ、拍子を更新したり、曲の終りを検出するサブルーチンのフローチャートである。

【図２４】演奏停止処理を行なうサブルーチンのフローチャートである。

【図２５】小節カウンタの曲頭の休符区間を検出し、楽譜通りの小節番号にするための処理を行なうサブルーチンのフローチャートである。

【図２６】小節番号を表示する処理を行なうサブルーチンのフローチャートである。

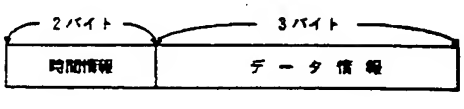
【図２７】従来の手法における演奏データのロード方法を示すフローチャートである。

【図２８】従来の手法の説明図である。

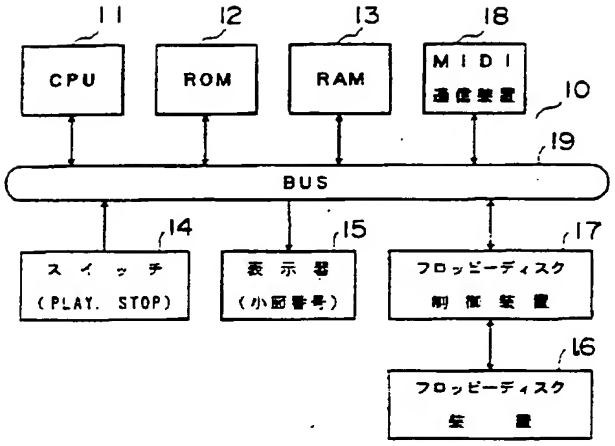
【図４】

トラック番号	track0		track1				track2		
そのセクタ内のクロック数	150	10	10	50	50	50	20	100	40
セクタ番号	sector0		sector1	sector2	sector3	sector4		sector5	sector6

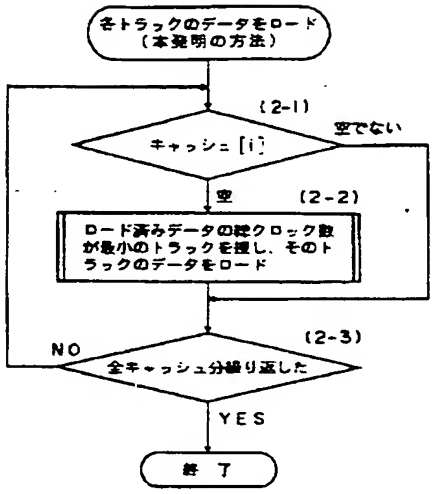
【図１６】



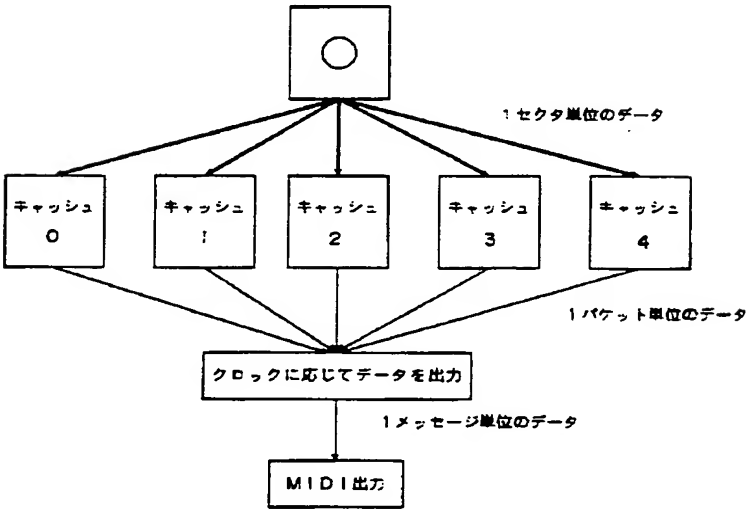
【図１】


11
12
13
18
CPU
ROM
RAM
MIDI
通信装置
10
19
BUS
14
15
17
スイッチ
(PLAY, STOP)
表示器
(小節番号)
フロッピーディスク
制御装置
16
フロッピーディスク
装置

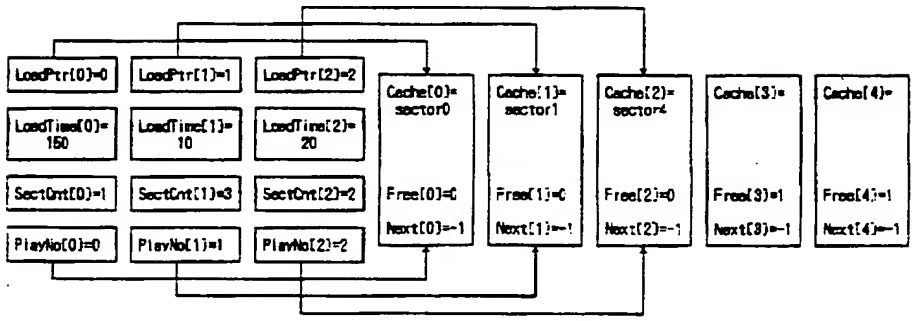
【図２】


各トラックのデータをロード
(本発明の方法)
(2-1)
キャッシュ[i]
空でない
空
(2-2)
ロード済みデータの経クロック数
が最小のトラックを捜し、そのト
ラックのデータをロード
(2-3)
NO
全キャッシュ分繰り返した
YES
終了

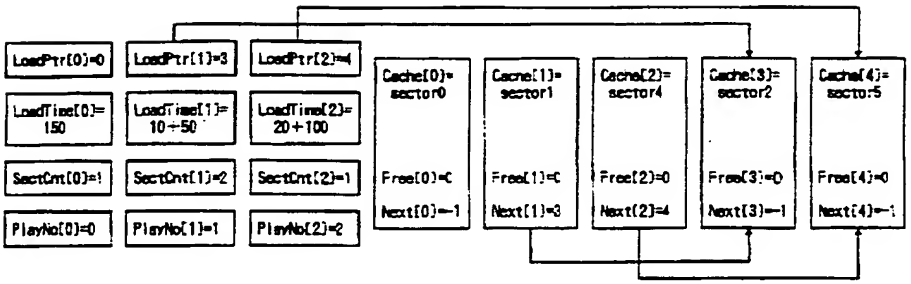
【図３】


1セクタ単位のデータ
キャッシュ
0
キャッシュ
1
キャッシュ
2
キャッシュ
3
キャッシュ
4
1パケット単位のデータ
クロックに応じてデータを出力
1メッセージ単位のデータ
MIDI出力

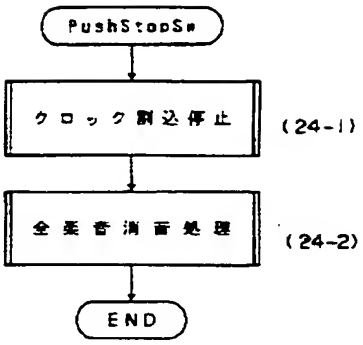
【図５】


LoadPtr[0]=0
LoadPtr[1]=1
LoadPtr[2]=2
LoadTime[0]=
150
LoadTime[1]=
10
LoadTime[2]=
20
SectCnt[0]=1
SectCnt[1]=3
SectCnt[2]=2
PlayNo[0]=0
PlayNo[1]=1
PlayNo[2]=2
Cache[0]=
sector0
Free[0]=0
Next[0]=-1
Cache[1]=
sector1
Free[1]=0
Next[1]=-1
Cache[2]=
sector4
Free[2]=0
Next[2]=-1
Cache[3]=
Free[3]=1
Next[3]=-1
Cache[4]=
Free[4]=1
Next[4]=-1

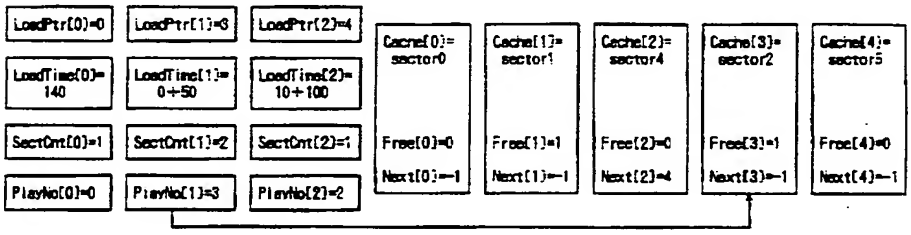
【図６】



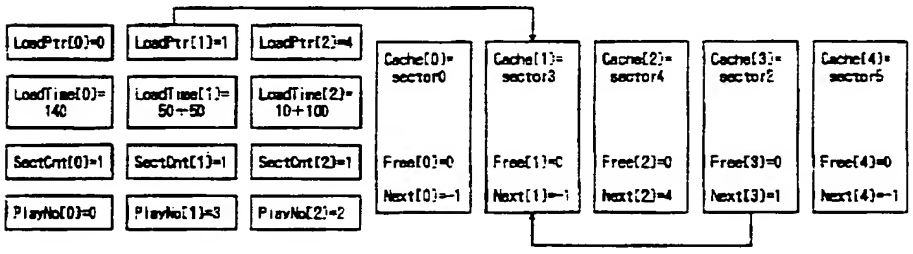
【図２４】



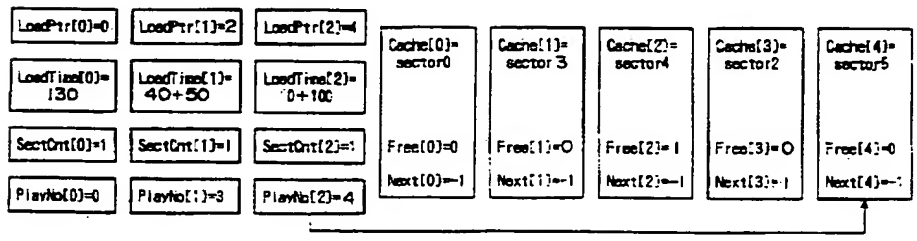
【図７】



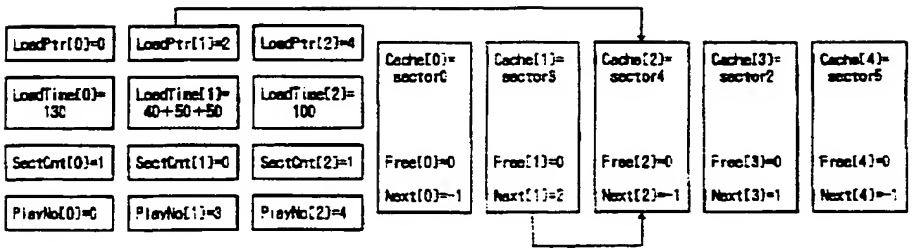
【図８】



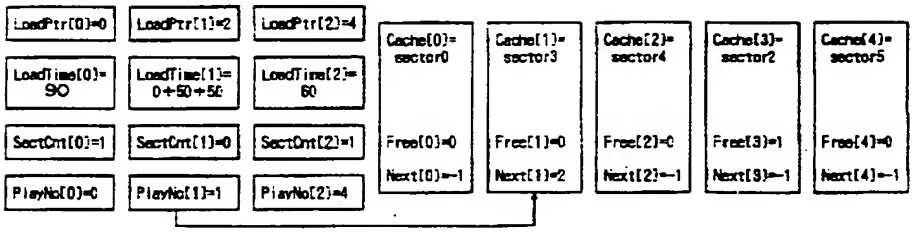
【図９】



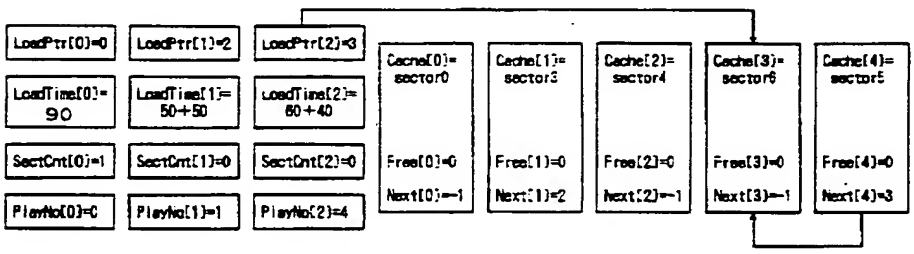
【図１０】



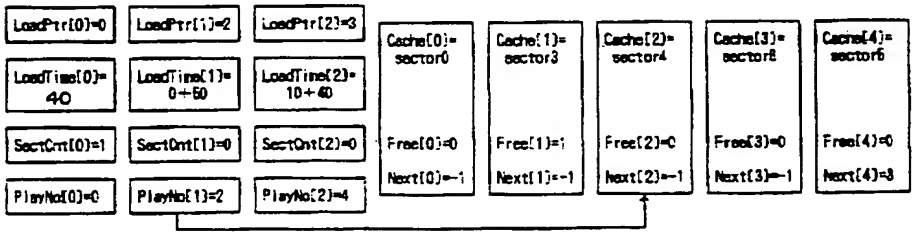
【図１１】



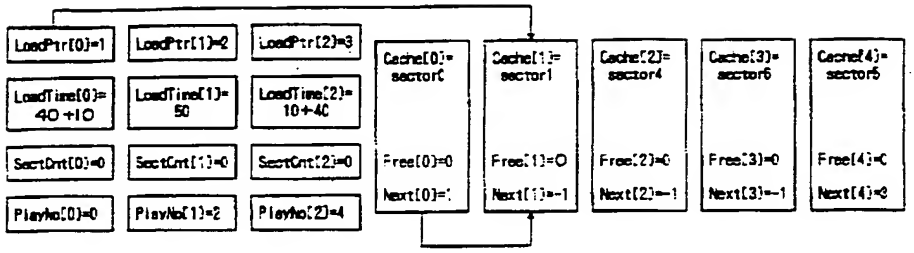
【図１２】



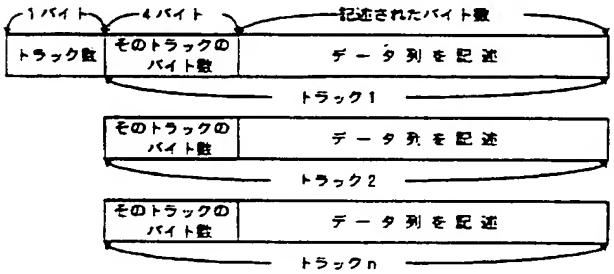
【図１３】



【図１４】

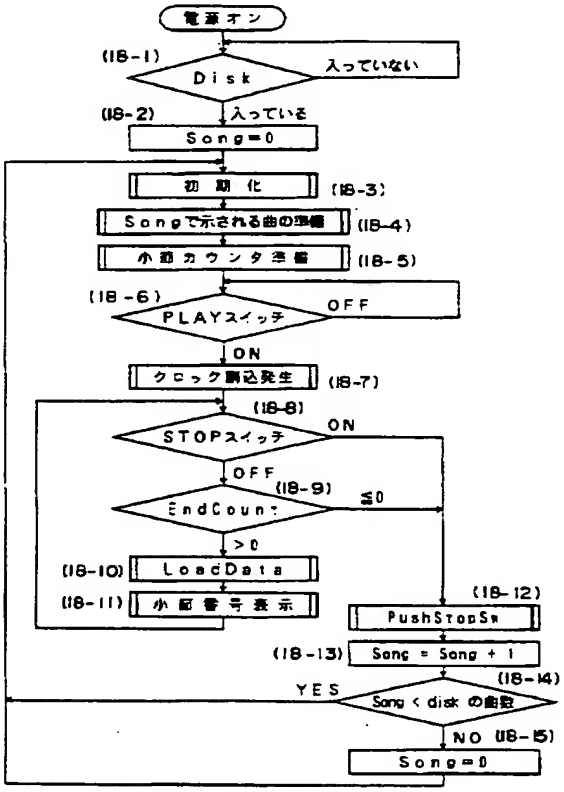


【図１５】

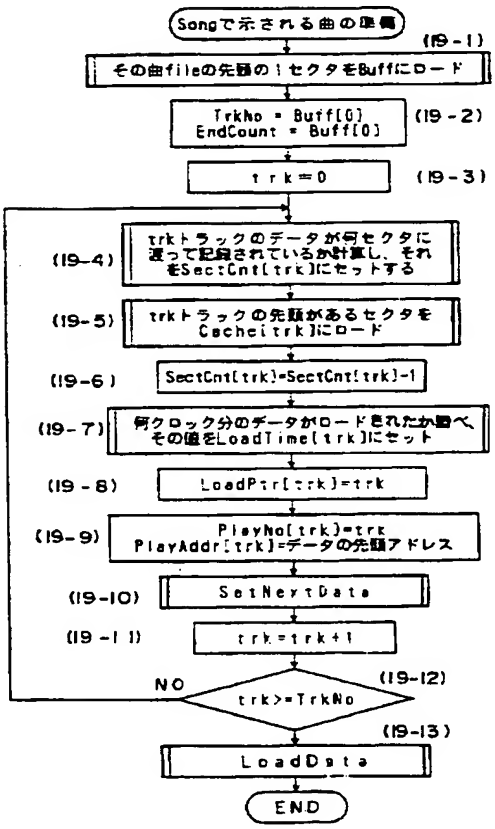


【図１７】

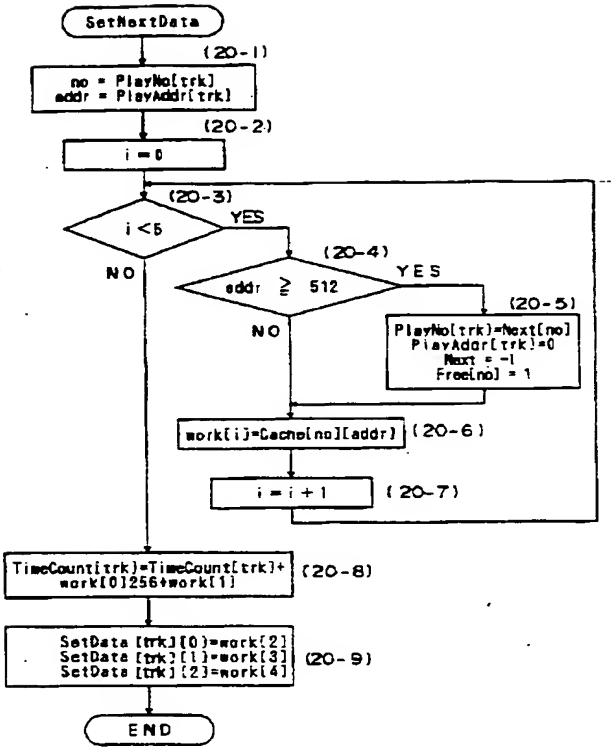
【図１８】



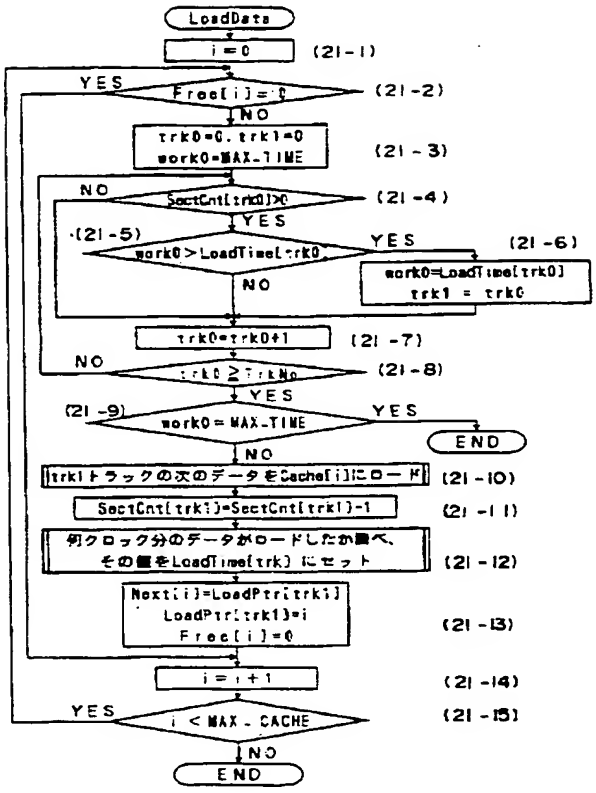
【図１９】



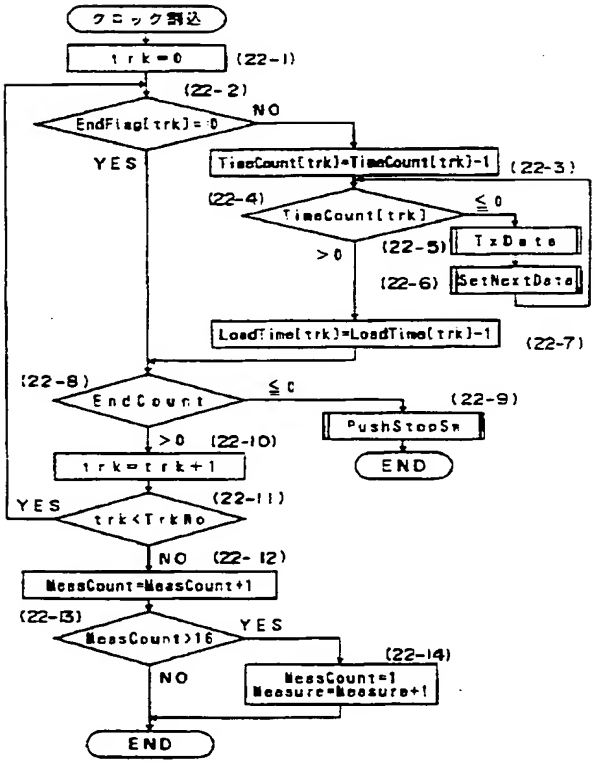
【図２０】



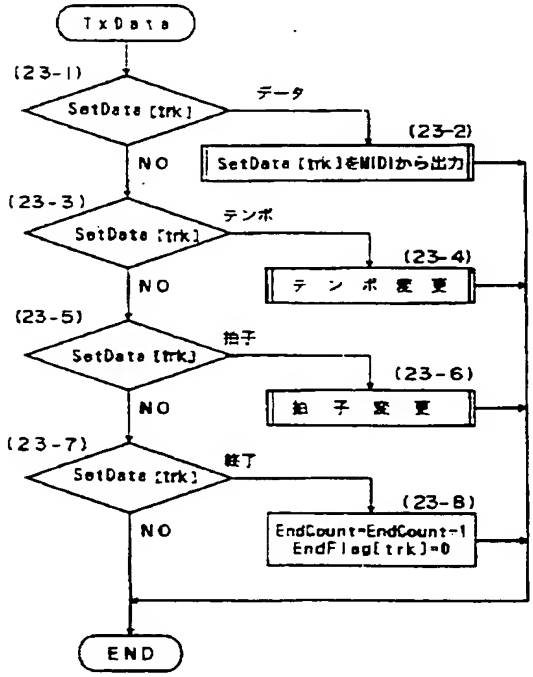
【図２１】



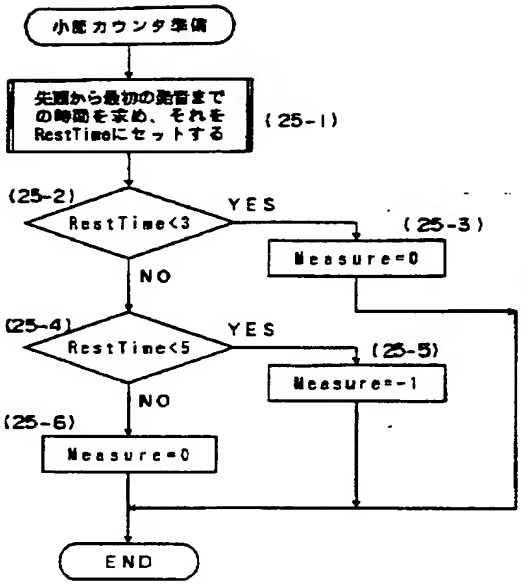
【図２２】



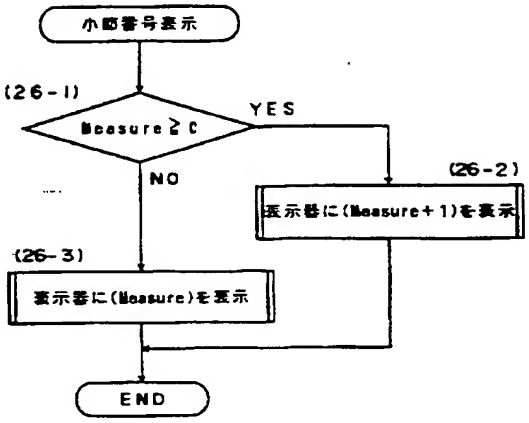
【図２３】



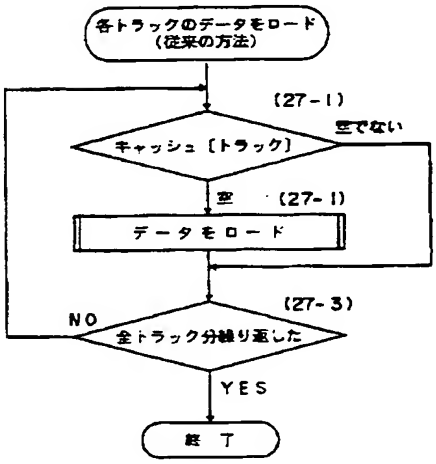
【図２５】



【図２６】



【図２７】



【図２８】